- 1 -

CODING OF MOTION VECTOR INFORMATION

RELATED APPLICATION INFORMATION

The following co-pending U.S. patent applications relate to the present

5 application and are hereby incorporated herein by reference: 1) U.S. Patent Application Serial No. aa/bbb,ccc, entitled, "Advanced Bi-Directional Predictive Coding of Video Frames," filed concurrently herewith; 2) U.S. Patent Application Serial No. aa/bbb,ccc, entitled, "Intraframe and Interframe Interlace Coding and Decoding," filed concurrently herewith; 3) U.S. Patent Application Serial No. 10/321,415, entitled, "Skip Macroblock

10 Coding," filed December 16, 2002; and 4) U.S. Patent Application Serial No. 10/379,615, entitled "Chrominance Motion Vector Rounding," filed March 4, 2003.

COPYRIGHT AUTHORIZATION

20 TECHNICAL FIELD

Techniques and tools for coding and decoding motion vector information are described. A video encoder uses an extended motion vector in a motion vector syntax for encoding predicted video frames.

25 BACKGROUND

Digital video consumes large amounts of storage and transmission capacity. A typical raw digital video sequence includes 15 or 30 frames per second. Each frame can include tens or hundreds of thousands of pixels (also called pels). Each pixel represents a tiny element of the picture. In raw form, a computer commonly represents

a pixel with 24 bits. Thus, the number of bits per second, or bit rate, of a typical raw digital video sequence can be 5 million bits/second or more.

Most computers and computer networks lack the resources to process raw digital video. For this reason, engineers use compression (also called coding or
5   encoding) to reduce the bit rate of digital video. Compression can be lossless, in which quality of the video does not suffer but decreases in bit rate are limited by the complexity of the video. Or, compression can be lossy, in which quality of the video suffers but decreases in bit rate are more dramatic. Decompression reverses compression.

10   In general, video compression techniques include intraframe compression and interframe compression. Intraframe compression techniques compress individual frames, typically called I-frames or key frames. Interframe compression techniques compress frames with reference to preceding and/or following frames, which are typically called predicted frames, P-frames, or B-frames.

15   Microsoft Corporation's Windows Media Video, Version 8 ["WMV8"] includes a video encoder and a video decoder. The WMV8 encoder uses intraframe and interframe compression, and the WMV8 decoder uses intraframe and interframe decompression.

20   **A.   Intraframe Compression in WMV8**

Figure 1 illustrates block-based intraframe compression 100 of a block 105 of pixels in a key frame in the WMV8 encoder. A block is a set of pixels, for example, an 8x8 arrangement of pixels. The WMV8 encoder splits a key video frame into 8x8 blocks of pixels and applies an 8x8 Discrete Cosine Transform ["DCT"] 110 to
25   individual blocks such as the block 105. A DCT is a type of frequency transform that converts the 8x8 block of pixels (spatial information) into an 8x8 block of DCT coefficients 115, which are frequency information. The DCT operation itself is lossless or nearly lossless.

The encoder then quantizes 120 the DCT coefficients, resulting in an 8x8 block
30   of quantized DCT coefficients 125. For example, the encoder applies a uniform, scalar

quantization step size to each coefficient. Quantization is lossy. The encoder then prepares the 8x8 block of quantized DCT coefficients 125 for entropy encoding, which is a form of lossless compression. The exact type of entropy encoding can vary depending on whether a coefficient is a DC coefficient (lowest frequency), an AC

5   coefficient (other frequencies) in the top row or left column, or another AC coefficient.

The encoder encodes the DC coefficient 126 as a differential from the DC coefficient 136 of a neighboring 8x8 block, which is a previously encoded neighbor (e.g., top or left) of the block being encoded. (Figure 1 shows a neighbor block 135 that is situated to the left of the block being encoded in the frame.) The encoder

10   entropy encodes 140 the differential.

The entropy encoder can encode the left column or top row of AC coefficients as a differential from a corresponding column or row of the neighboring 8x8 block. Figure 1 shows the left column 127 of AC coefficients encoded as a differential 147 from the left column 137 of the neighboring (to the left) block 135. The differential

15   coding increases the chance that the differential coefficients have zero values. The remaining AC coefficients are from the block 125 of quantized DCT coefficients.

The encoder scans 150 the 8x8 block 145 of predicted, quantized AC DCT coefficients into a one-dimensional array 155 and then entropy encodes the scanned AC coefficients using a variation of run length coding 160. The encoder selects an

20   entropy code from one or more run/level/last tables 165 and outputs the entropy code.

B.   Interframe Compression in WMV8

Interframe compression in the WMV8 encoder uses block-based motion compensated prediction coding followed by transform coding of the residual error.

25   Figures 2 and 3 illustrate the block-based interframe compression for a predicted frame in the WMV8 encoder. In particular, Figure 2 illustrates motion estimation for a predicted frame 210 and Figure 3 illustrates compression of a prediction residual for a motion-estimated block of a predicted frame.

For example, the WMV8 encoder splits a predicted frame into 8x8 blocks of

30   pixels. Groups of four 8x8 blocks form macroblocks. For each macroblock, a motion

estimation process is performed. The motion estimation approximates the motion of the macroblock of pixels relative to a reference frame, for example, a previously coded, preceding frame. In Figure 2, the WMV8 encoder computes a motion vector for a macroblock 215 in the predicted frame 210. To compute the motion vector, the

5   encoder searches in a search area 235 of a reference frame 230. Within the search area 235, the encoder compares the macroblock 215 from the predicted frame 210 to various candidate macroblocks in order to find a candidate macroblock that is a good match. After the encoder finds a good matching macroblock, the encoder outputs information specifying the motion vector (entropy coded) for the matching macroblock

10   so the decoder can find the matching macroblock during decoding. When decoding the predicted frame 210 with motion compensation, a decoder uses the motion vector to compute a prediction macroblock for the macroblock 215 using information from the reference frame 230. The prediction for the macroblock 215 is rarely perfect, so the encoder usually encodes 8x8 blocks of pixel differences (also called the error or

15   residual blocks) between the prediction macroblock and the macroblock 215 itself.

Figure 3 illustrates an example of computation and encoding of an error block 335 in the WMV8 encoder. The error block 335 is the difference between the predicted block 315 and the original current block 325. The encoder applies a DCT 340 to the error block 335, resulting in an 8x8 block 345 of coefficients. The encoder then

20   quantizes 350 the DCT coefficients, resulting in an 8x8 block of quantized DCT coefficients 355. The quantization step size is adjustable. Quantization results in loss of precision, but not complete loss of the information for the coefficients.

The encoder then prepares the 8x8 block 355 of quantized DCT coefficients for entropy encoding. The encoder scans 360 the 8x8 block 355 into a one dimensional

25   array 365 with 64 elements, such that coefficients are generally ordered from lowest frequency to highest frequency, which typically creates long runs of zero values.

The encoder entropy encodes the scanned coefficients using a variation of run length coding 370. The encoder selects an entropy code from one or more run/level/last tables 375 and outputs the entropy code.

- 5 -

Figure 4 shows an example of a corresponding decoding process 400 for an inter-coded block. Due to the quantization of the DCT coefficients, the reconstructed block 475 is not identical to the corresponding original block. The compression is lossy.

5   In summary of Figure 4, a decoder decodes (410, 420) entropy-coded information representing a prediction residual using variable length decoding 410 with one or more run/level/last tables 415 and run length decoding 420. The decoder inverse scans 430 a one-dimensional array 425 storing the entropy-decoded information into a two-dimensional block 435. The decoder inverse quantizes and

10   inverse discrete cosine transforms (together, 440) the data, resulting in a reconstructed error block 445. In a separate motion compensation path, the decoder computes a predicted block 465 using motion vector information 455 for displacement from a reference frame. The decoder combines 470 the predicted block 465 with the reconstructed error block 445 to form the reconstructed block 475.

15   The amount of change between the original and reconstructed frame is termed the distortion and the number of bits required to code the frame is termed the rate for the frame. The amount of distortion is roughly inversely proportional to the rate. In other words, coding a frame with fewer bits (greater compression) will result in greater distortion, and vice versa.

20

C.   Bi-directional Prediction

Bi-directionally coded images (e.g., B-frames) use two images from the source video as reference (or anchor) images. For example, referring to Figure 5, a B-frame 510 in a video sequence has a temporally previous reference frame 520 and a

25   temporally future reference frame 530.

Some conventional encoders use five prediction modes (forward, backward, direct, interpolated and intra) to predict regions in a current B-frame. In intra mode, an encoder does not predict a macroblock from either reference image, and therefore calculates no motion vectors for the macroblock. In forward and backward modes, an

30   encoder predicts a macroblock using either the previous or future reference frame, and

therefore calculates one motion vector for the macroblock. In direct and interpolated

modes, an encoder predicts a macroblock in a current frame using both reference

frames. In interpolated mode, the encoder explicitly calculates two motion vectors for

the macroblock. In direct mode, the encoder derives *implied* motion vectors by scaling

5   the co-located motion vector in the future reference frame, and therefore does not

explicitly calculate any motion vectors for the macroblock.

D.   Interlace Coding

A typical interlace video frame consists of two fields scanned at different times.

10   For example, referring to Figure 6, an interlace video frame 600 includes top field 610

and bottom field 620. Typically, the odd-numbered lines (top field) are scanned at one

time (e.g., time t) and the even-numbered lines (bottom field) are scanned at a different

(typically later) time (e.g., time $t + 1$). This arrangement can create jagged tooth-like

features in regions of a frame where motion is present because the two fields are

15   scanned at different times. On the other hand, in stationary regions, image structures

in the frame may be preserved (i.e., the interlace artifacts visible in motion regions may

not be visible in stationary regions). Macroblocks in interlace frames can be field-

coded or frame-coded. In field-coded macroblocks, the top-field lines and bottom-field

lines are rearranged, such that the top field lines appear at the top of the macroblock,

20   and the bottom field lines appear at the bottom of the macroblock. Predicted field-

coded macroblocks typically have one motion vector for each field in the macroblock.

In frame-coded macroblocks, the field lines alternate between top-field lines and

bottom-field lines. Predicted frame-coded macroblocks typically have one motion

vector for the macroblock.

25

E.   Standards for Video Compression and Decompression

Aside from WMV8, several international standards relate to video compression

and decompression. These standards include the Motion Picture Experts Group

["MPEG"] 1, 2, and 4 standards and the H.261, H.262, and H.263 standards from the

International Telecommunication Union ["ITU"]. Like WMV8, these standards use a combination of intraframe and interframe compression.

For example, advanced video compression or encoding techniques (including techniques in the MPEG, H.26x and WMV8 standards) are based on the exploitation of

5   temporal coherence of typical video sequences. Image areas are tracked as they move over time, and information pertaining to the motion of these areas is compressed as part of the bit stream. Traditionally, a standard P-frame is encoded by computing and storing motion information in the form of two-dimensional displacement vectors corresponding to regularly-sized image tiles (e.g, macroblocks) For example, a

10 macroblock may have one motion vector (a 1MV macroblock) for the macroblock or a motion vector for each of four blocks in the macroblock (a 4MV macroblock). Subsequently, the difference between the input frame and its motion compensated prediction is compressed, usually in a suitable transform domain, and added to an encoded bit stream. Typically, the motion vector component of the bitstream makes up

15 between 10% and 30% of the size. Therefore, it can be appreciated that efficient motion vector coding is a key factor in efficient video compression.

Motion vector coding efficiency can be achieved in different ways. For example, motion vectors are often highly correlated between neighboring macroblocks. For efficiency, a motion vector of a given macroblock can be differentially coded from

20 its prediction based on a causal neighborhood of adjacent macroblocks. A few exceptions to this general rule are observed in prior algorithms, such as those described in MPEG-4 and WMV8:

1   When the predicted motion vector lies outside a certain area (typically ±16 pixels from zero, for either component), the prediction is pulled back to the

25   nearest point within this area.

2   When the vectors making up the causal neighborhood of the current macroblock are diverse (e.g., at motion discontinuities), the "Hybrid Motion Vector" mode is employed – the prediction is signaled by a codeword that indicates whether to use the motion vector to the top or to the left (or any other

30   combination).

3   When a macroblock is essentially unchanged from its reference frame (i.e., a (0, 0) motion vector (no motion) and no residual components), it is indicated as being "skipped."

4   A macroblock may be coded as intra (i.e., not differentially predicted from the previous frame). In this case, no motion vector is sent. (Otherwise, for non-skipped macroblocks that are not intra coded, a motion vector is always sent.)

5   Intra coded macroblocks are indicated by an "I/P switch", which is jointly coded with a coded block pattern (or CBP). The CBP indicates which of the blocks making up a macroblock have attached residual information.

Given the critical importance of video compression and decompression to digital video, it is not surprising that video compression and decompression are richly developed fields. Whatever the benefits of previous video compression and decompression techniques, however, they do not have the advantages of the following techniques and tools.

SUMMARY

In summary, the detailed description is directed to various techniques and tools for encoding and decoding motion vector information for video images. The various techniques and tools can be used in combination or independently.

In one aspect, a video encoder jointly codes for a set of pixels (e.g., block, macroblock, etc.) a switch code with motion vector information (e.g., a motion vector for an inter-coded block/macroblock, or a pseudo motion vector for an intra-coded block/macroblock). The switch code indicates whether a set of pixels is intra-coded.

In another aspect, a video encoder yields an extended motion vector code by jointly coding for a set of pixels a switch code, motion vector information, and a terminal symbol indicating whether subsequent data is encoded for the set of pixels. The subsequent data can include coded block pattern data and/or residual data for macroblocks. The extended motion vector code can be included in an alphabet or table of codes. In one aspect, the alphabet lacks a code that would represent a skip condition for the set of pixels.

In another aspect, an encoder/decoder selects motion vector predictors for current macroblocks (e.g., 1MV or mixed 1MV/4MV macroblocks) in a video image (e.g., an interlace or progressive P-frame or B-frame).

For example, an encoder/decoder selects a predictor from a set of candidates
5   for a last macroblock of a macroblock row. The set of candidates comprises motion vectors from a set of macroblocks adjacent to the current macroblock. The set of macroblocks adjacent to the current macroblock consists of a top adjacent macroblock, a left adjacent macroblock, and a top-left adjacent macroblock. The predictor can be a motion vector for an individual block within a macroblock.

10   As another example, an encoder/decoder selects a predictor from a set of candidates comprising motion vectors from a set of blocks in macroblocks adjacent to a current macroblock. The set of blocks consists of a bottom-left block of a top adjacent macroblock, a top-right block of a left adjacent macroblock, and a bottom-right block of a top-left adjacent macroblock.

15   As another example, an encoder/decoder selects a predictor for a current top-left block in the first macroblock of a macroblock row from a set of candidates. The set of candidates comprises a zero-value motion vector and motion vectors from a set of blocks in an adjacent macroblock. The set of blocks consists of a bottom-left block of a top adjacent macroblock, and a bottom-right block of the top adjacent macroblock.

20   As another example, an encoder/decoder selects a predictor for a current top-right block of a current macroblock from a set of candidates. The current macroblock is the last macroblock of a macroblock row, and the set of candidates consists of a motion vector from the top-left block of the current macroblock, a motion vector from a bottom-left block of a top adjacent macroblock, and a motion vector from a bottom-right
25   block of the top adjacent macroblock.

In another aspect, a video encoder/decoder calculates a motion vector predictor for a set of pixels (e.g., a 1MV or mixed 1MV/4MV macroblock) based on analysis of candidates, and compares the calculated predictor with one or more of the candidates (e.g., the left and top candidates). Based on the comparison, the encoder/decoder
30   determines whether to replace the calculated motion vector predictor with a hybrid

motion vector of one of the candidates. The set of pixels can be a skipped set of pixels (e.g., a skipped macroblock). The hybrid motion vector can be indicated by an indicator bit.

5   In another aspect, a video encoder/decoder selects a motion vector mode for a predicted image from a set of modes comprising a mixed one- and four-motion vector, quarter-pixel resolution, bicubic interpolation filter mode; a one-motion vector, quarter-pixel resolution, bicubic interpolation filter mode; a one-motion vector, half-pixel resolution, bicubic interpolation filter mode; and a one-motion vector, half-pixel resolution, bilinear interpolation filter mode. The mode can be signaled in a bit stream

10   at various levels (e.g., frame-level, slice-level, group-of-pictures level, etc.). The set of modes also can include other modes, such as a four-motion vector, 1/8-pixel, six-tap interpolation filter mode.

    In another aspect, for a set of pixels, a video encoder finds a motion vector component value and a motion vector predictor component value, each within a

15   bounded range. The encoder calculates a differential motion vector component value (which is outside the bounded range) based on the motion vector component value and the motion vector predictor component value. The encoder represents the differential motion vector component value with a signed binary code in a bit stream. The signed binary code is operable to allow reconstruction of the differential motion vector

20   component value. For example, the encoder performs rollover arithmetic to convert the differential motion vector component value into a signed binary code. The number of bits in the signed binary code can vary based on motion data (e.g., motion vector component direction (x or y), motion vector resolution, motion vector range.

    In another aspect, a video decoder decodes a set of pixels in an encoded bit

25   stream by receiving an extended motion vector code for the set of pixels. The extended motion vector code reflects joint encoding of motion information together with information indicating whether the set of pixels is intra-coded or inter-coded and with a terminal symbol. The decoder determines whether subsequent data for the set of pixels is included in the encoded bit stream based on the extended motion vector code

30   (e.g., by the terminal symbol in the code). For a macroblocks (e.g., 4:2:0, 4:1:1, or

4:2:2 macroblocks), subsequent data can include a coded block pattern code and/or residual information for one or more blocks in the macroblock.

In the bit stream, the extended motion vector code can be preceded by, for example, header information or a modified coded block pattern code, and can be followed by other information for the set of pixels, such as a coded block pattern code. The decoder can receive more than one extended motion vector code for a set of pixels. For example, the decoder can receive two such codes for a bi-directionally predicted, or field-coded interlace macroblock. Or, the decoder can receive an extended motion vector code for each block in a macroblock.

In another aspect, a computer system includes means for decoding images, which comprises means for receiving an extended motion vector code and means for determining whether subsequent data for the set of pixels is included in the encoded bit stream based at least in part upon the received extended motion vector code.

In another aspect, a computer system includes means for encoding images, which comprises means for sending an extended motion vector code for a set of pixels as part of an encoded bit stream.

Additional features and advantages will be made apparent from the following detailed description of different embodiments that proceeds with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a diagram showing block-based intraframe compression of an 8x8 block of pixels according to the prior art.

Figure 2 is a diagram showing motion estimation in a video encoder according to the prior art.

Figure 3 is a diagram showing block-based interframe compression for an 8x8 block of prediction residuals in a video encoder according to the prior art.

Figure 4 is a diagram showing block-based interframe decompression for an 8x8 block of prediction residuals in a video encoder according to the prior art.

- 12 -

Figure 5 is a diagram showing a B-frame with past and future reference frames according to the prior art.

Figure 6 is a diagram showing an interlaced video frame according to the prior art.

5   Figure 7 is a block diagram of a suitable computing environment in which several described embodiments may be implemented.

Figure 8 is a block diagram of a generalized video encoder system used in several described embodiments.

Figure 9 is a block diagram of a generalized video decoder system used in 10   several described embodiments.

Figure 10 is a diagram showing a macroblock syntax with an extended motion vector symbol for use in coding progressive 1MV macroblocks in P-frames, forward/backward predicted macroblocks in B-frames, and interlace frame-type macroblocks.

15   Figure 11 is a diagram showing a macroblock syntax with an extended motion vector symbol for use in coding progressive 4MV macroblocks in P-frames.

Figure 12 is a diagram showing a macroblock syntax with extended motion vector symbols for use in coding progressive interpolated macroblocks in B-frames, forward/backward predicted macroblocks in B-frames, and interlace frame-type 20   macroblocks.

Figure 13 is a diagram showing a macroblock syntax with extended motion vector symbols for use in coding interlace macroblocks in P-frames and forward/backward predicted field-type macroblocks in B-frames.

Figure 14 is a diagram showing a macroblock syntax with extended motion 25   vector symbols for use in coding interlace interpolated field-type macroblocks in B-frames.

Figure 15 is a diagram showing a macroblock comprising four blocks.

Figures 16A and 16B are diagrams showing candidate motion vector predictors for a 1MV macroblock in a P-frames.

- 13 -

Figures 17A and 17B are diagrams showing candidate motion vector predictors for a 1MV macroblock in a mixed 1MV/4MV P-frame.

Figures 18A and 18B are diagrams showing candidate motion vector predictors for a block at position 0 in a 4MV macroblock in a mixed 1MV/4MV P-frame.

5   Figures 19A and 19B are diagrams showing candidate motion vector predictors for a block at position 1 in a 4MV macroblock in a mixed 1MV/4MV P-frame.

Figure 20 is a diagram showing candidate motion vector predictors for a block at position 2 in a 4MV macroblock in a mixed 1MV/4MV P-frame.

Figure 21 is a diagram showing candidate motion vector predictors for a block

10   at position 3 in a 4MV macroblock in a mixed 1MV/4MV P-frame.

Figures 22A and 22B are diagrams showing candidate motion vector predictors for a frame-type macroblock in an interlace P-frame.

Figures 23A and 23B are diagrams showing candidate motion vector predictors for a field-type macroblock in an interlace P-frame.

15   Figure 24 is a flow chart showing a technique for performing a pull back for a motion vector predictor.

Figure 25 is a flow chart showing a technique for determining whether to use a hybrid motion vector for a set of pixels.

Figure 26 is a flow chart showing a technique for applying rollover arithmetic to

20   a differential motion vector.

DETAILED DESCRIPTION

The present application relates to techniques and tools for coding motion information in video image sequences. Bit stream formats or syntaxes include flags

25   and other codes to incorporate the techniques. Different bit stream formats can comprise different layers or levels (e.g., sequence level, frame/picture/image level, macroblock level, and/or block level).

The various techniques and tools can be used in combination or independently. Different embodiments implement one or more of the described techniques and tools.

30

I.   Computing Environment

Figure 7 illustrates a generalized example of a suitable computing environment 700 in which several of the described embodiments may be implemented. The computing environment 700 is not intended to suggest any limitation as to scope of use

5   or functionality, as the techniques and tools may be implemented in diverse general-purpose or special-purpose computing environments.

With reference to Figure 7, the computing environment 700 includes at least one processing unit 710 and memory 720. In Figure 7, this most basic configuration 730 is included within a dashed line. The processing unit 710 executes computer-

10   executable instructions and may be a real or a virtual processor. In a multi-processing system, multiple processing units execute computer-executable instructions to increase processing power. The memory 720 may be volatile memory (e.g., registers, cache, RAM), non-volatile memory (e.g., ROM, EEPROM, flash memory, etc.), or some combination of the two. The memory 720 stores software 780 implementing a video

15   encoder or decoder.

A computing environment may have additional features. For example, the computing environment 700 includes storage 740, one or more input devices 750, one or more output devices 760, and one or more communication connections 770. An interconnection mechanism (not shown) such as a bus, controller, or network

20   interconnects the components of the computing environment 700. Typically, operating system software (not shown) provides an operating environment for other software executing in the computing environment 700, and coordinates activities of the components of the computing environment 700.

The storage 740 may be removable or non-removable, and includes magnetic

25   disks, magnetic tapes or cassettes, CD-ROMs, DVDs, or any other medium which can be used to store information and which can be accessed within the computing environment 700. The storage 740 stores instructions for the software 780 implementing the video encoder or decoder.

The input device(s) 750 may be a touch input device such as a keyboard,

30   mouse, pen, or trackball, a voice input device, a scanning device, or another device

that provides input to the computing environment 700. For audio or video encoding, the input device(s) 750 may be a sound card, video card, TV tuner card, or similar device that accepts audio or video input in analog or digital form, or a CD-ROM or CD-RW that reads audio or video samples into the computing environment 700. The

5   output device(s) 760 may be a display, printer, speaker, CD-writer, or another device that provides output from the computing environment 700.

The communication connection(s) 770 enable communication over a communication medium to another computing entity. The communication medium conveys information such as computer-executable instructions, audio or video input or

10   output, or other data in a modulated data signal. A modulated data signal is a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media include wired or wireless techniques implemented with an electrical, optical, RF, infrared, acoustic, or other carrier.

15   The techniques and tools can be described in the general context of computer-readable media. Computer-readable media are any available media that can be accessed within a computing environment. By way of example, and not limitation, with the computing environment 700, computer-readable media include memory 720, storage 740, communication media, and combinations of any of the above.

20   The techniques and tools can be described in the general context of computer-executable instructions, such as those included in program modules, being executed in a computing environment on a target real or virtual processor. Generally, program modules include routines, programs, libraries, objects, classes, components, data structures, etc. that perform particular tasks or implement particular abstract data

25   types. The functionality of the program modules may be combined or split between program modules as desired in various embodiments. Computer-executable instructions for program modules may be executed within a local or distributed computing environment.

For the sake of presentation, the detailed description uses terms like "predict,"

30   "choose," "compensate," and "apply" to describe computer operations in a computing

environment. These terms are high-level abstractions for operations performed by a computer, and should not be confused with acts performed by a human being. The actual computer operations corresponding to these terms vary depending on implementation.

5

II.   Generalized Video Encoder and Decoder

Figure 8 is a block diagram of a generalized video encoder 800 and Figure 9 is a block diagram of a generalized video decoder 900.

The relationships shown between modules within the encoder and decoder

10   indicate the main flow of information in the encoder and decoder; other relationships are not shown for the sake of simplicity. In particular, Figures 8 and 9 generally do not show side information indicating the encoder settings, modes, tables, etc. used for a video sequence, frame, macroblock, block, etc. Such side information is sent in the output bit stream, typically after entropy encoding of the side information. The format

15   of the output bit stream can be a Windows Media Video format or another format.

The encoder 800 and decoder 900 are block-based and use a 4:2:0 macroblock format with each macroblock including four 8x8 luminance blocks and two 8x8 chrominance blocks, or a 4:1:1 macroblock format with each macroblock including four 8x8 luminance blocks and four 4x8 chrominance blocks. Alternatively, the encoder 800

20   and decoder 900 are object-based, use a different macroblock or block format, or perform operations on sets of pixels of different size or configuration.

Depending on implementation and the type of compression desired, modules of the encoder or decoder can be added, omitted, split into multiple modules, combined with other modules, and/or replaced with like modules. In alternative embodiments,

25   encoder or decoders with different modules and/or other configurations of modules perform one or more of the described techniques.

A.   Video Encoder

Figure 8 is a block diagram of a general video encoder system 800. The

30   encoder system 800 receives a sequence of video frames including a current frame

805, and produces compressed video information 895 as output. Particular
embodiments of video encoders typically use a variation or supplemented version of
the generalized encoder 800.

The encoder system 800 compresses predicted frames and key frames. For
the sake of presentation, Figure 8 shows a path for key frames through the encoder
system 800 and a path for predicted frames. Many of the components of the encoder
system 800 are used for compressing both key frames and predicted frames. The
exact operations performed by those components can vary depending on the type of
information being compressed.

A predicted frame (also called P-frame, B-frame, or inter-coded frame) is
represented in terms of prediction (or difference) from one or more reference (or
anchor) frames. A prediction residual is the difference between what was predicted
and the original frame. In contrast, a key frame (also called I-frame, intra-coded frame)
is compressed without reference to other frames.

If the current frame 805 is a forward-predicted frame, a motion estimator 810
estimates motion of macroblocks or other sets of pixels of the current frame 805 with
respect to a reference frame, which is the reconstructed previous frame 825 buffered in
a frame store (e.g., frame store 820). If the current frame 805 is a bi-directionally-
predicted frame (a B-frame), a motion estimator 810 estimates motion in the current
frame 805 with respect to two reconstructed reference frames. Typically, a motion
estimator estimates motion in a B-frame with respect to a temporally previous
reference frame and a temporally future reference frame. Accordingly, the encoder
system 800 can comprise separate stores 820 and 822 for backward and forward
reference frames. For more information on bi-directionally predicted frames, see U.S.
Patent Application Serial No. aa/bbb,ccc, entitled, "Advanced Bi-Directional Predictive
Coding of Video Frames," filed concurrently herewith.

The motion estimator 810 can estimate motion by pixel, ½ pixel, ¼ pixel, or
other increments, and can switch the resolution of the motion estimation on a frame-by-
frame basis or other basis. The resolution of the motion estimation can be the same or
different horizontally and vertically. The motion estimator 810 outputs as side

- 18 -

information motion information 815 such as motion vectors. A motion compensator 830 applies the motion information 815 to the reconstructed frame(s) 825 to form a motion-compensated current frame 835. The prediction is rarely perfect, however, and the difference between the motion-compensated current frame 835 and the original

5   current frame 805 is the prediction residual 845. Alternatively, a motion estimator and motion compensator apply another type of motion estimation/compensation.

A frequency transformer 860 converts the spatial domain video information into frequency domain (i.e., spectral) data. For block-based video frames, the frequency transformer 860 applies a discrete cosine transform ["DCT"] or variant of DCT to blocks

10   of the pixel data or prediction residual data, producing blocks of DCT coefficients. Alternatively, the frequency transformer 860 applies another conventional frequency transform such as a Fourier transform or uses wavelet or subband analysis. If the encoder uses spatial extrapolation (not shown in Figure 8) to encode blocks of key frames, the frequency transformer 860 can apply a re-oriented frequency transform

15   such as a skewed DCT to blocks of prediction residuals for the key frame. In some embodiments, the frequency transformer 860 applies an 8x8, 8x4, 4x8, or other size frequency transforms (e.g., DCT) to prediction residuals for predicted frames.

A quantizer 870 then quantizes the blocks of spectral data coefficients. The quantizer applies uniform, scalar quantization to the spectral data with a step-size that

20   varies on a frame-by-frame basis or other basis. Alternatively, the quantizer applies another type of quantization to the spectral data coefficients, for example, a non-uniform, vector, or non-adaptive quantization, or directly quantizes spatial domain data in an encoder system that does not use frequency transformations. In addition to adaptive quantization, the encoder 800 can use frame dropping, adaptive filtering, or

25   other techniques for rate control.

If a given macroblock in a predicted frame has no information of certain types (e.g., no motion information for the macroblock and/or no residual information), the encoder 800 may encode the macroblock as a skipped macroblock. If so, the encoder signals the skipped macroblock in the output bit stream of compressed video

30   information 895.

- 19 -

When a reconstructed current frame is needed for subsequent motion estimation/compensation, an inverse quantizer 876 performs inverse quantization on the quantized spectral data coefficients. An inverse frequency transformer 866 then performs the inverse of the operations of the frequency transformer 860, producing a

5   reconstructed prediction residual (for a predicted frame) or a reconstructed key frame. If the current frame 805 was a key frame, the reconstructed key frame is taken as the reconstructed current frame (not shown). If the current frame 805 was a predicted frame, the reconstructed prediction residual is added to the motion-compensated current frame 835 to form the reconstructed current frame. A frame store (e.g., frame

10   store 820) buffers the reconstructed current frame for use in predicting another frame. In some embodiments, the encoder applies a deblocking filter to the reconstructed frame to adaptively smooth discontinuities in the blocks of the frame.

The entropy coder 880 compresses the output of the quantizer 870 as well as certain side information (e.g., motion information 815, spatial extrapolation modes,

15   quantization step size). Typical entropy coding techniques include arithmetic coding, differential coding, Huffman coding, run length coding, LZ coding, dictionary coding, and combinations of the above. The entropy coder 880 typically uses different coding techniques for different kinds of information (e.g., DC coefficients, AC coefficients, different kinds of side information), and can choose from among multiple code tables

20   within a particular coding technique.

The entropy coder 880 puts compressed video information 895 in the buffer 890. A buffer level indicator is fed back to bit rate adaptive modules.

The compressed video information 895 is depleted from the buffer 890 at a constant or relatively constant bit rate and stored for subsequent streaming at that bit

25   rate. Therefore, the level of the buffer 890 is primarily a function of the entropy of the filtered, quantized video information, which affects the efficiency of the entropy coding. Alternatively, the encoder system 800 streams compressed video information immediately following compression, and the level of the buffer 890 also depends on the rate at which information is depleted from the buffer 890 for transmission.

- 20 -

Before or after the buffer 890, the compressed video information 895 can be channel coded for transmission over the network. The channel coding can apply error detection and correction data to the compressed video information 895.

5   **B.   Video Decoder**

Figure 9 is a block diagram of a general video decoder system 900. The decoder system 900 receives information 995 for a compressed sequence of video frames and produces output including a reconstructed frame 905. Particular embodiments of video decoders typically use a variation or supplemented version of

10   the generalized decoder 900.

The decoder system 900 decompresses predicted frames and key frames. For the sake of presentation, Figure 9 shows a path for key frames through the decoder system 900 and a path for predicted frames. Many of the components of the decoder system 900 are used for decompressing both key frames and predicted frames. The

15   exact operations performed by those components can vary depending on the type of information being decompressed.

A buffer 990 receives the information 995 for the compressed video sequence and makes the received information available to the entropy decoder 980. The buffer 990 typically receives the information at a rate that is fairly constant over time, and

20   includes a jitter buffer to smooth short-term variations in bandwidth or transmission. The buffer 990 can include a playback buffer and other buffers as well. Alternatively, the buffer 990 receives information at a varying rate. Before or after the buffer 990, the compressed video information can be channel decoded and processed for error detection and correction.

25   The entropy decoder 980 entropy decodes entropy-coded quantized data as well as entropy-coded side information (e.g., motion information 915, spatial extrapolation modes, quantization step size), typically applying the inverse of the entropy encoding performed in the encoder. Entropy decoding techniques include arithmetic decoding, differential decoding, Huffman decoding, run length decoding, LZ

30   decoding, dictionary decoding, and combinations of the above. The entropy decoder

980 frequently uses different decoding techniques for different kinds of information (e.g., DC coefficients, AC coefficients, different kinds of side information), and can choose from among multiple code tables within a particular decoding technique.

5   A motion compensator 930 applies motion information 915 to one or more reference frames 925 to form a prediction 935 of the frame 905 being reconstructed. For example, the motion compensator 930 uses a macroblock motion vector to find a macroblock in a reference frame 925. A frame buffer (e.g., frame buffer 920) stores previously reconstructed frames for use as reference frames. Typically, B-frames have more than one reference frame (e.g., a temporally previous reference frame and a

10   temporally future reference frame). Accordingly, the decoder system 900 can comprise separate frame buffers 920 and 922 for backward and forward reference frames.

The motion compensator 930 can compensate for motion at pixel, ½ pixel, ¼ pixel, or other increments, and can switch the resolution of the motion compensation on a frame-by-frame basis or other basis. The resolution of the motion compensation

15   can be the same or different horizontally and vertically. Alternatively, a motion compensator applies another type of motion compensation. The prediction by the motion compensator is rarely perfect, so the decoder 900 also reconstructs prediction residuals.

When the decoder needs a reconstructed frame for subsequent motion

20   compensation, a frame buffer (e.g., frame buffer 920) buffers the reconstructed frame for use in predicting another frame. In some embodiments, the decoder applies a deblocking filter to the reconstructed frame to adaptively smooth discontinuities in the blocks of the frame.

An inverse quantizer 970 inverse quantizes entropy-decoded data. In general,

25   the inverse quantizer applies uniform, scalar inverse quantization to the entropy-decoded data with a step-size that varies on a frame-by-frame basis or other basis. Alternatively, the inverse quantizer applies another type of inverse quantization to the data, for example, a non-uniform, vector, or non-adaptive quantization, or directly inverse quantizes spatial domain data in a decoder system that does not use inverse

30   frequency transformations.

- 22 -

An inverse frequency transformer 960 converts the quantized, frequency domain data into spatial domain video information. For block-based video frames, the inverse frequency transformer 960 applies an inverse DCT ["IDCT"] or variant of IDCT to blocks of the DCT coefficients, producing pixel data or prediction residual data for

5   key frames or predicted frames, respectively. Alternatively, the frequency transformer 960 applies another conventional inverse frequency transform such as a Fourier transform or uses wavelet or subband synthesis. If the decoder uses spatial extrapolation (not shown in Figure 9) to decode blocks of key frames, the inverse frequency transformer 960 can apply a re-oriented inverse frequency transform such

10   as a skewed IDCT to blocks of prediction residuals for the key frame. In some embodiments, the inverse frequency transformer 960 applies an 8x8, 8x4, 4x8, or other size inverse frequency transforms (e.g., IDCT) to prediction residuals for predicted frames.

When a skipped macroblock is signaled in the bit stream of information 995 for

15   a compressed sequence of video frames, the decoder 900 reconstructs the skipped macroblock without using information (e.g., motion information and/or residual information) normally included in the bit stream for non-skipped macroblocks.

III.   Overview of Motion Vector Coding

20   The described techniques and tools improve compression efficiency for predicted images (e.g., frames) in video sequences. Described techniques and tools apply to a one-motion-vector-per-macroblock (1MV) model of motion estimation and compensation for predicted frames (e.g., P-frames). Described techniques and tools also employ specialized mechanisms to encode motion vectors in certain situations

25   (e.g., four-motion-vectors-per-macroblock (4MV) models, mixed 1MV and 4MV models, B-frames, and interlace coding) that give rise to data structures that are not homogeneous with the 1MV model. For more information on interlace video, see U.S. Patent Application Serial No. aa/bbb,ccc, entitled, "Intraframe and Interframe Interlace Coding and Decoding," filed concurrently herewith. Described techniques and tools

30   are also extensible to future formats.

With an increased average number of motion vectors per frame (e.g., in 4MV and mixed 1MV and 4MV models), it is desirable to design a more efficient scheme to encode motion vector information. As in earlier standards, described techniques and tools use predictive coding to compress motion vector information. However, there are

5   several key differences. The described techniques and tools, individually or in combination, include the following features:

1. An extended motion vector alphabet:

   a. The I/P switch is jointly coded with the motion vector. In other words, a bit code indicating that a macroblock (or block) is to be coded as an

10   intra macroblock or intra block, respectively, is joint coded with a pseudo motion vector, the joint code indicating it is an intra macroblock/block.

   b. In addition to the I/P switch, a "terminal" symbol is coded jointly with the motion vector. The terminal symbol indicates whether there is any subsequent data pertaining to the object (macroblock, block, etc.) being

15   coded. The joint symbol is referred to as an extended motion vector ("MV*").

2. A sub-frame-level (e.g., macroblock level) syntax using an extended motion vector alphabet to efficiently code, e.g., progressive 1MV macroblocks, 4MV macroblocks and B-frames, and interlace 1MV macroblocks, 2MV macroblocks

20   and B-frames.

3. Generation of motion vector predictors and differential motion vectors.

4. Hybrid motion vector encoding with different criteria for identifying hybrid motion vectors.

5. Efficient signaling of motion vector modes at frame level.

25 6. Differential coding of motion vector residuals based on rollover arithmetic, (similar to modulo arithmetic) to avoid need for pull-back of predictors.

These features are explained in detail in the following sections.

In some embodiments, an encoder derives motion vectors for chrominance planes from luminance motion vectors. However, the techniques and tools described

30   herein are equally applicable to chrominance motion in other embodiments. For

- 24 -

example, a video encoder may choose to explicitly send chrominance motion vectors as part of a bit stream, and can use techniques and tools similar to those described herein to encode/decode the chrominance motion vectors.

5   **IV.   Extended Motion Vector Alphabet**

In some embodiments, an extended motion vector alphabet includes joint codes for jointly coding motion vector information with other information for a block, macroblock, or other set of pixels.

10   **A.   Signaling Intra Macroblocks and Blocks**

The signaling of an intra-coded set of pixels (e.g., block, macroblock, etc.) can be achieved by extending the alphabet of motion vectors to allow for a symbol (e.g., an I/P switch) indicating an intra area. Intra macroblocks and blocks do not have a true motion vector associated with them. A motion vector (or in the case of an intra-coded

15   set of pixels, a pseudo motion vector) can be appended to an intra symbol to yield a triple of the form <Intra, MVx, MVy> that indicates whether the set of pixels (e.g., macroblock or block) is coded as intra, and if not, what its motion vector should be. When the intra flag is set, MVx and MVy are "don't care" conditions. When the intra flag is zero, MVx and MVy correspond to computed motion vector components.

20   Joint coding of an intra symbol with motion vectors allows an elegant yet efficient implementation with the ability to switch blocks to intra when four extended motion vectors are used in a macroblock.

      B.   Signaling Residual Information

25   In addition to the intra symbol, some embodiments jointly code the presence or absence of subsequent residual symbols with a motion vector. For example, a "last" (or terminal) symbol indicates whether the joint code containing the motion vector or pseudo motion vector is a terminal symbol of a given macroblock, block or field, or if residual data follows (e.g., when last = 1 (i.e. last is true), no subsequent data pertains

30   to the area). This joint code can be referred to as an extended motion vector, and is of

- 25 -

the form <intra, MVx, MVy, last>. In the syntax diagrams below, an extended motion vector is represented as MV*.

In some embodiments, the extended motion vector symbol <inter, 0, 0, true> is an invalid symbol. The condition that would ordinarily lead to this symbol a special

5   condition called a "skip" condition. Under the skip condition, the current set of pixels (e.g., macroblock) can be predicted (to within quantization error) from its motion vector. No additional data (e.g., residual data) is necessary to decode this area. For efficiency reasons, the skip condition can signaled at the frame level. Therefore, in some embodiments, this symbol is not present in the bit stream. For example, skipped

10   macroblocks have a motion vector such that the *differential* motion vector is (0, 0) or have no motion at all. In other words, in skipped macroblocks where some motion is present, the skipped macroblocks use the same motion vector as the predicted motion vector. Skipped macroblocks are also defined for 4MV macroblocks, and other cases. For more information on skipped macroblocks, see U.S. Patent Application Serial No.

15   10/321,415, entitled, "Skip Macroblock Coding," filed December 16, 2002.

The last symbol applies to both intra signals and inter motion vectors. The way this symbol is used in different embodiments depends on many factors, including whether a macroblock is a 1MV or 4MV macroblock, or an interlace macroblock (e.g., a field-coded, 2MV macroblock). Moreover, in some embodiments, the last symbol is

20   interpreted differently for interpolated mode B-frames. These concepts are covered in detail below.

V.   Syntax for Coding Motion Vector Information

In some embodiments, a video encoder encodes video images using a sub-

25   frame-level syntax (e.g., a macroblock-level syntax) including extended motion vectors. For example, for macroblocks in a video sequence having progressive and interlace P-frames and B-frames, each macroblock is coded with zero, one, two or four associated extended motion vector symbols. The specific number of motion vectors depends on the specifics of the coding mode – (e.g., whether the frame is a P-frame or B-frame, progressive or interlace, 1MV or 4MV-coded, and/or skip coded). Coding modes also

30

determine the order in which the motion vector information is sent. The following sections and corresponding Figures 10-14 cover these possibilities and map out the syntax or format for different situations. Although the figures show elements (e.g., extended motion vectors) in certain arrangements, the elements can be arranged in

5   different ways.

In the following sections and the corresponding figures, the symbol MBH denotes a macroblock header -- a placeholder for any macroblock level information other than a motion vector, I/P switch or coded block pattern (CBP)). Examples of elements in MBH are skip bit information, motion vector mode information, coding

10   mode information for B-frames, and frame/field information for interlace frames.

A.   1MV Macroblock Syntax

Figure 10 is a diagram showing an exemplary macroblock syntax 1000 with an extended motion vector symbol for use in coding 1MV macroblocks. Examples of 1MV

15   macroblocks include progressive P-frame macroblocks, interlace frame-coded P-frame macroblocks, progressive forward- or backward-predicted B-frame macroblocks, and interlace frame-coded forward- or backward-predicted B-frame macroblocks. In Figure 10, MV* is sent after MBH and before CBP.

CBP indicates which of the blocks making up a macroblock have attached

20   residual information. For example, for a 4:2:0 macroblock with four luminance blocks and two chrominance blocks, CBP includes six bits. A corresponding CBP bit indicates whether residual information exists for each block. In MV*, the terminal symbol "last" is set to 1 if CBP is all zero, indicating that there are no residuals for all six blocks in the macroblock. In this case, CBP is not sent. If CBP is not all zero (which under many

25   circumstances is more likely to be the case), the terminal symbol is set to 1, and the CBP is sent, followed by the residual data for blocks that have residuals. For example, in Figure 10, up to six residual blocks (e.g., luminance residual blocks Y0, Y1, Y2, and Y3, and chrominance residual blocks U and V) can be sent, depending on the value of CBP.

30

B.   4MV Macroblock Syntax

Figure 11 is a diagram showing an exemplary macroblock syntax 1100 with an extended motion vector symbol for use in coding progressive 4MV macroblocks in P-frames. For the code labeled CBP', when four motion vectors are present in a

5   macroblock, the first four components of the CBP (corresponding to the first four blocks) are reinterpreted to be the union of the events where MV* ≠ 0, and where residuals are present. For example, in Figure 11, the first four CBP components correspond to the luminance blocks. When a luminance block is intra-coded or inter-coded with a nonzero differential motion vector, or when there are residuals, the block

10   pattern is set to true. There is no change to the chrominance components.

In Figure 11, the CBP is sent right after MBH. Subsequently, the extended motion vectors for the four luminance blocks are sent only when the corresponding block pattern is nonzero. The terminal symbols of the extended motion vectors are used to send the original CBP information for the luminance blocks, flagging the

15   presence of residuals. As an illustration, if block Y0 has no residuals but does have a nonzero differential motion vector, the first component of CBP would normally be set to true. Therefore, MV* is sent, with its last symbol being set to true. No further information is sent for block Y0.

20   ### C.   2MV Macroblock Syntax

Figure 12 is a diagram showing an exemplary macroblock syntax 1200 with extended motion vector symbols for use in coding 2MV macroblocks (e.g., progressive interpolated macroblocks in B-frames, forward/backward predicted macroblocks in B-frames, and interlace frame-type macroblocks). For example, in progressive

25   sequences and in frame coded interlace sequences, B-frame macroblocks use zero, one or two motion vectors. When there are two motion vectors, the syntax 1200 shown in Figure 12 is used. This is an extension of the 1MV macroblock syntax 1100 shown in Figure 11.

In Figure 12, the two extended motion vectors MV1* and MV2* are sent in a

30   predetermined order. For example, in some embodiments, an encoder sends a

- 28 -

backward differential motion vector followed by a forward differential motion vector for a B-frame macroblock, following the macroblock header. In the event that all residuals are zero, the last symbol of the second motion vector is set to true and no further data is sent. In the event that MV2* = 0 and CBP = 0, the last symbol of MV1* is set to true

5   and the macroblock terminates. When both motion vectors and CBP are zero, the macroblock is skip-coded.

D.   Macroblock Syntax for Interlace Field-type Macroblocks in P-frames and Forward/Backward Predicted Field-type Macroblocks in

10   **B-frames**

Figure 13 is a diagram showing an exemplary macroblock syntax 1300 with extended motion vector symbols for use in coding interlace field-type macroblocks in P-frames and forward/backward predicted field-type macroblocks in B-frames. Such macroblocks have two motion vectors, corresponding to the top and bottom field

15   motion. The extended motion vectors are sent subsequent to a modified CBP (CBP' in Figure 13). The first and third components of the CBP are reinterpreted to be the union of the corresponding nonzero extended motion vector events and nonzero residual events. The terminal symbols of the top extended motion vector MVT* and the bottom extended motion vector MVB* contain the original block pattern components for the

20   corresponding blocks. Although Figure 13 shows the extended motion vectors in certain locations, other arrangements are also valid.

E.   Macroblock Syntax for Interlace Field-type Interpolated Macroblocks in B-frames

25   Figure 14 is a diagram showing an exemplary macroblock syntax with extended motion vector symbols for use in coding interlace interpolated (bi-directional) field-type macroblocks in B-frames. The technique used to code motion vectors for interlace field-type interpolated B-frame macroblocks combines ideas from interlace field-type P-frame macroblocks and progressive B-frame macroblocks using 2 motion vectors.

30   Again, while Figure 14 shows an exemplary arrangement having certain overloaded

- 29 -

CBP blocks, the four extended motion vectors (e.g., MV1T*, MV2T*, MV1B* and MV2B*) can be distributed differently across the block data channels.

F.   Simplified CBP and MV* Alphabets

5   In the syntax formats described above, the coded block pattern CBP = 0 (i.e., all bits in CBP are equal to zero) does not occur in the bit stream. Accordingly, in some embodiments, for the sake of efficiency, this symbol is not present in the CBP alphabet. For example, for the six blocks in a 4:2:0 macroblock, the coded block pattern alphabet comprises $2^6 - 1 = 63$ symbols. Moreover, as discussed earlier, the

10   MV* symbol <intra switch, MVx, MVy, last> = <inter, 0, 0, true> is an invalid symbol. Occurrences of this symbol can be coded using skip bits, or in some cases, CBP.

VI.   Generation of Motion Vector Predictors and Differential Motion Vectors

            In some embodiments, to exploit continuity in motion vector information, motion

15   vectors are differentially predicted and encoded from neighboring sets of pixels (e.g., blocks, macroblocks, etc.). For example, a video encoder/decoder uses three motion vectors in the neighborhood of a current block, macroblock or field for computing a prediction. The specific features of a predictor calculation technique depend on factors such as whether the sequence is interlace or progressive, and whether one, two, or

20   four motion vectors are being generated for a given macroblock. For example, in a 1MV macroblock, the macroblock has one corresponding motion vector for the entire macroblock. In a 4MV macroblock, the macroblock has one corresponding motion vector for each block in the macroblock. Figure 15 is a diagram showing a macroblock 1500 comprising four blocks, the macroblock 1500 has a motion vector corresponding

25   to each block in positions 0-3.

            In the following sections, there is only one numerical prediction for a given motion vector, and this is calculated by analyzing candidates (which may also be referred to as predictors) for the motion vector predictor.

A.   Motion Vector Candidates in 1MV P-frames

Figures 16A and 16B are diagrams showing three candidate motion vector predictors for a current 1MV macroblock 1610 in a P-frame. In Figure 16A, where the current macroblock 1610 is not the last macroblock in a macroblock row, the

5   candidates are taken from the left (Predictor C), top (Predictor A) and top-right (Predictor B) macroblocks. In Figure 16B, the macroblock 1610 is the last macroblock in the row. In this case, Predictor B is taken from the top-left macroblock instead of the top-right. In some embodiments, for the special case where the frame is one macroblock wide, the predictor is always Predictor A (the top predictor).

10

B.   Motion Vector Candidates in Mixed-MV P-frames

Figures 17A, 17B, 18A, 18B, 19A, 19B, 20 and 21 show candidate motion vector predictors for 1MV and 4MV macroblocks in mixed-MV P-frames. In these figures, the larger squares are macroblock boundaries and the smaller squares are

15   block boundaries. In some embodiments, for the special case where the frame is one macroblock wide, the predictor is always Predictor A (the top predictor).

Figures 17A and 17B are diagrams showing candidate motion vector predictors for a 1MV macroblock 1710 in a mixed 1MV/4MV P-frame. The neighboring macroblocks may be 1MV or 4 MV macroblocks. Figures 17A and 17B show the

20   candidate motion vectors under an assumption that the neighbors are 4MV macroblocks. For example, Predictor A is the motion vector for block 2 in the macroblock above the current macroblock 1710 and Predictor C is the motion vector for block 1 in the macroblock immediately to the left of the current macroblock 1710. If any of the neighbors are 1MV macroblocks, the motion vector predictors shown in

25   Figures 17A and 17B are taken to be the motion vectors for the entire neighboring macroblock. As Figure 17B shows, if the macroblock 1710 is the last macroblock in the row, then Predictor B is from block 3 of the top-left macroblock instead of from block 2 in the top-right macroblock (as in Figure 17A).

In embodiments such as those shown in Figures 17A and 17B, Predictor B is

30   taken from the adjacent macroblock column instead of the block immediately to the

right of Predictor A because, in the case where the top macroblock (in which Predictor A lies) is 1MV-coded, the block adjacent to Predictor A will have the same motion vector as A. This can essentially force the predictor to predict from the top, which is not always desirable.

5   Figures 18A, 18B, 19A, 19B, 20 and 21 show predictors for each of the 4 luminance blocks in a 4MV macroblock. For example, Figures 18A and 18B are diagrams showing candidate motion vector predictors for a block 1810 at position 0 in a 4MV macroblock 1820 in a mixed 1MV/4MV P-frame. In some embodiments, for the case where the macroblock 1820 is the first macroblock in the row, Predictor B for

10   block 1810 is handled differently than the remaining blocks in the row. In Figure 18B, Predictor B is taken from the block at position 3 in the macroblock immediately above the current macroblock 1820 instead of from the block at position 3 in the macroblock above and to the left of current macroblock 1820, as is the case in Figure 18A. Again, in some embodiments, Predictor B is to the left of Predictor A in the more frequently

15   occurring case shown in Figure 18A because the block to the immediate right of Predictor A will have the same motion vector as Predictor A when the top macroblock is 1MV-coded. In Figure 18B, Predictor C is equal to zero because it lies outside the picture boundary.

     Figures 19A and 19B are diagrams showing candidate motion vector predictors

20   for a block 1910 at position 1 in a 4MV macroblock 1920 in a mixed 1MV/4MV P-frame. In Figure 19B, for the case where the macroblock 1920 is the last macroblock in the row, Predictor B for the current block 1910 is handled differently than for the case shown in Figure 19A. In Figure 19B, Predictor B is taken from the block at position 2 in the macroblock immediately above the current macroblock 1920 instead of from the

25   block at position 2 in the macroblock above and to the left of the current macroblock 1920, as is the case in Figure 19A.

     Figure 20 is a diagram showing candidate motion vector predictors for a block 2010 at position 2 in a 4MV macroblock 2020 in a mixed 1MV/4MV P-frame. In Figure 20, if the macroblock 2020 is in the first macroblock column (in other words, if the

macroblock 2020 is the first macroblock in a macroblock row) then Predictor C for the blocks 2010 is equal to zero.

Figure 21 is a diagram showing candidate motion vector predictors for a block 2110 at position 3 in a 4MV macroblock 2120 in a mixed 1MV/4MV P-frame. The predictors for block 2110 are the three other blocks within the macroblock 2120. The choice for Predictor B to be taken from the block to the left of Predictor A (e.g., instead of the block to the right of Predictor A) is for causality. In situations such as the example shown in Figure 21, the block 2110 can be decoded without referencing motion vector information from a subsequent macroblock.

C.   Motion Vector Candidates in Interlace P-frames

Figures 22A and 22B are diagrams showing candidate motion vector predictors for a frame-type macroblock 2210 in an interlace P-frame. In Figure 22A, where the current macroblock 2210 is not the last macroblock in a macroblock row, the candidates are taken from the left (Predictor C), top (Predictor A) and top-right (Predictor B) macroblocks. In Figure 22B, the macroblock 2210 is the last macroblock in the row. In this case, Predictor B is taken from the top-left macroblock instead of the top-right. In some embodiments, for the special case where the frame is one macroblock wide, the predictor is always Predictor A (the top predictor). When a neighboring macroblock is field-coded, having two motion vectors (one for the top field and the other for the bottom field), the two motion vectors are averaged to generate the prediction candidate. The figure below shows how the motion vector predictor is derived from the neighboring macroblocks for a frame coded macroblock in Interlace P pictures.

In some embodiments, for field-coded macroblocks, the motion vectors of corresponding fields of the neighboring macroblocks are used as candidates for predicting a motion vector for a top or bottom field. For example, Figures 23A and 23B are diagrams showing candidate motion vector predictors for a field-type macroblock 2310 in an interlace P-frame. In Figure 23A, where the current macroblock 2310 is not the last macroblock in a macroblock row, the candidates are taken from fields in the left

(Predictor C), top (Predictor A) and top-right (Predictor B) macroblocks. In Figure 23B, the macroblock 2310 is the last macroblock in the row. In this case, Predictor B is taken from the top-left macroblock instead of the top-right. When a neighboring macroblock is frame coded, the motion vectors corresponding to its fields are deemed to be equal to the motion vector for the entire macroblock. In other words, the top and bottom motion vectors are set to V, where V is the motion vector of the entire macroblock.

D.   Calculating a Predictor from Candidates

Given three motion vector predictor candidates, the following pseudocode illustrates the process for calculating the motion vector predictor.

```
if (predictorA is not out of bound) {
        if (predictorC is out of bound && predictorB is out of bound) {
                // picture consists of one MB
                predictor = predictorA;
        } else {
                if (predictorC is out of bound) {
                        predictorC = 0;
                }
                numIntra = 0;
                if (predictorA is intra) {
                        predictorA  = 0;
                        numIntra = numIntra + 1;
                }
                if (predictorB is intra) {
                        predictorB  = 0;
                        numIntra = numIntra + 1;
                }
                if (predictorC is intra) {
                        predictorC = 0;
```

- 34 -

```
            numIntra = numIntra + 1;
        }


        // calculate predictor from A, B and C predictor candidates
        predictor = cmedian3(predictorA, predictorB, predictorC);
    }
} else if (predictorC is not out of bound) {
    predictor = predictorC;
} else {
    predictor  = 0;
}
```

The function cmedian3 is the component-wise median of three two dimensional vectors.

E.   Pullback of Predictor

In some embodiments, after the predictor is computed, an encoder/decoder verifies whether the area of the image referenced by the predictor is within the frame. If the area is entirely outside the frame, it is pulled back to an area that overlaps the frame by one pixel width, overlapping the frame at the area closest to the original area. For example, Figure 24 shows a technique 24 for performing a pull back for a motion vector predictor. At 2410, an encoder/decoder calculates a predictor. At 2420, the encoder/decoder then finds the area referenced by the calculated predictor. At 2430, the encoder/decoder determines whether the referenced area is completely outside the frame. If not, the process ends. If so, the encoder/decoder at 2440 pulls back the predictor.

In some embodiments, an encoder/decoder uses the following rules for performing predictor pull backs:

1. For a macroblock motion vector: The top-left point of a 16x16 area pointed to by the predictor is restricted to be from −15 to (picture width − 1) in the vertical and horizontal dimensions.

2. For a block motion vector: The top-left point of a 8x8 area pointed to by the predictor is restricted to be from −7 to (picture width − 1) in the vertical and horizontal dimensions.

3. For a field motion vector: In the horizontal dimension, the top-left point of a

5   8x16 area pointed to by the predictor is restricted to be from −15 to (picture width − 1). In the vertical dimension, the top-left point of this area is restricted to be from −7 to (picture height − 1).

Although the predicted motion vector prior to pullback is valid, pullback assures that more diversity is available in the local area around the predictor. This allows for better

10   predictions by lowering the cost of useful motion vectors.

F.   Hybrid Motion Vectors

        In some embodiments, if a P-frame is 1MV or mixed-MV, a calculated predictor is tested relative to the A and C predictors, such as those described above. This test

15   determines whether the motion vector must be hybrid coded.

        For example, Figure 25 is a flow chart showing a technique 2500 for determining whether to use a hybrid motion vector for a set of pixels (e.g., a macroblock, block, etc.). At 2510, a video encoder/decoder calculates a predictor for a set of pixels. At 2520, the encoder/decoder compares the calculated predictor to one

20   or more predictor candidates. At 2530, the encoder/decoder determines whether a hybrid motion vector should be used. If not, the encoder/decoder at 2540 uses the previously calculated predictor to predict the motion vector for the set of pixels. If so, the encoder/decoder at 2550 uses a hybrid motion indicator to determine or signal which candidate predictor to use as the predictor for the set of pixels.

25   When the variance among the three motion vector candidates used in a prediction is high, the true motion vector is likely to be close to one of the candidate vectors, especially the vectors to the left and the top of the current macroblock or block (Predictors A and C, respectively). When the candidates are far apart, their component-wise median is often not an accurate predictor of motion in a current

30   macroblock. Hence, in some embodiments, an encoder sends an additional bit

indicating which candidate the true motion vector is closer to. For example, when the indicator bit indicates that the motion vector for Predictor A or C is the closer one, a decoder uses it as the predictor. The decoder must determine for each motion vector whether to expect a hybrid motion indicator bit, and this determination can be made

5   from causal motion vector information.

The following pseudo-code illustrates this determination. In this example, when either Predictor A or Predictor C is intra-coded, the corresponding motion is deemed to be zero.

predictor: The calculated motion vector prediction, possibly reset below

10   sabs(): Sum of absolute values of components

```
if ((predictorA is out of bounds) || (predictorC is out of bounds))
{
        return 0 //not a hybrid motion vector
}
else
{
        if (predictorA is intra)
                sum = sabs(predictor)
        else
                sum = abs(predictor - predictorA)
        if (sum > 32)
                return 1 // hybrid motion vector
        else
        {
                if (predictorC is intra)
                        sum = sabs(predictor)
                else
                        sum = abs(predictor - predictorC)
                if (sum > 32)
```

```
                     return 1 // hybrid motion vector

          }
                 return 0 // not a hybrid motion vector

     }
```

5   An advantage of the above approach is that it uses the computed predictor – and in the typical case when there is no hybrid motion, the additional computations are not expensive.

In some embodiments, in a bit stream syntax, the hybrid motion vector indicator bit is sent together with the motion vector itself. Hybrid motion vectors may occur even

10   when a set of pixels (e.g., block, macroblock, etc.) is skipped, in which case the one bit indicates whether to use A or C as the true motion for the set of pixels. In such cases, in the bit stream syntax, the hybrid bit is sent where the motion vector would have been had it not been skipped.

Hybrid motion vector prediction can be enabled or disabled in different

15   situations. For example, in some embodiments, hybrid motion vector prediction is not used for interlace pictures (e.g., field-coded P pictures). A decision to use hybrid motion vector prediction can be made at frame level, sequence level, or some other level.

20   **VII.   Motion Vector Modes**

In some embodiments, motion vectors are specified to half-pixel or quarter-pixel accuracy. Frames can also be 1MV frames, or mixed 1MV/4MV frames, and can use bicubic or bilinear interpolation. These choices make up the motion vector mode. In some embodiments, the motion vector mode is sent at the frame level. Alternatively,

25   an encoder chooses motion vector modes on some other basis, and/or sends motion vector mode information at some other level.

In some embodiments, an encoder uses one of four motion compensation modes. The frame-level mode indicates (a) possible number of motion vectors per macroblock, (b) motion vector sampling accuracy, and (c) interpolation filter. The four

30   modes (ranked in order of complexity / overhead cost) are:

1. Mixed 1MV/4MV per macroblock, quarter pixel, bicubic interpolation

2. 1MV per macroblock, quarter pixel, bicubic interpolation

3. 1MV per macroblock, half pixel, bicubic interpolation

4. 1MV per macroblock, half pixel, bilinear interpolation

VIII. Motion Vector Range and Rollover Arithmetic

Some embodiments use motion vectors that are specified in dyadic (power of two) ranges, with the range of permissible motion vectors in the x-component being larger than the range in the y-component. The range in the x-component is generally larger because (a) high motion typically occurs in the horizontal direction and (b) the cost of motion compensation with a large displacement is typically much higher in the vertical direction.

Some embodiments specify a baseline motion vector range of -64 to 63.x pixels for the x-component, and -32 to 31.x pixels for the y-component. The ".x" fraction is dependent on motion vector resolution. For example, for half-pixel sampling, .x is 0.5 and for quarter-pixel accuracy .x is 0.75. The total number of discrete motion vector components in the x and y directions are therefore 512 and 256, respectively, for bicubic filters (for bilinear filters, these numbers are 256 and 128). In other embodiments, the range is expanded to allow longer motion vectors in "broadcast modes."

Table 1 shows different ranges for motion vectors (in addition to the baseline), signaled by the variable-length codeword MVRANGE.

MVRANGE	Range in X	Range in Y
0 (baseline)	(-64, 63.x)	(-32, 31.x)
10	(-128, 127.x)	(-64, 63.x)
110	(-512, 511.x)	(-128, 127.x)
111	(-1024, 1023.x)	(-256, 255.x)

Table 1: Extended motion vector range

Motion vectors are transmitted in the bit stream by encoding their differences from causal predictors. Since the ranges of both motion vectors and predictors are bounded (e.g., by one of the ranges described above), the range of the differences is also bounded. In order to maximize encoding efficiency, rollover arithmetic is used to
5   encode the motion vector difference.

Figure 26 shows a technique 2600 for applying rollover arithmetic to a differential motion vector. For example, at 2610, an encoder finds a motion vector component for a macroblock. The encoder then finds a predictor for that motion vector component at 2620. At 2630, the encoder calculates a differential for the motion vector
10 component, based on the predictor. At 2640, the encoder then applies rollover arithmetic to encode the differential. Motion vector encoding using rollover arithmetic on the differential motion vector is a computationally simple yet efficient solution.

Let the operation Rollover(I, K) convert I into a signed K bit representation such that the lower K bits of I match those of Rollover(I, K). We know the following: If A and
15 B are integers, or fixed point numbers, such that Rollover(A, K) = A and Rollover(B, K) = B, then:

B = Rollover(A + Rollover(B − A, K), K).

Replacing A with MVPx and B with MVx, the following relationship holds:

MVx = Rollover(MVPx + Rollover(MVx − MVPx), K)

20 where K is chosen as the logarithm to base 2 of the motion vector alphabet size, assuming the size is a power of 2. The differential motion vector ΔMVx is set to Rollover(MVx − MVPx), which is represented in K bits.

In some embodiments, rollover arithmetic is applied according to the following example.
25 Assume that the current frame is encoded using the baseline motion vector range, with quarter pixel accuracy motion vectors. The range of both the x-component of a motion vector of a macroblock (MVx) and the x-component of its predicted motion (MVPx) is (-64, 63.75). The alphabet size for each is 2^9 = 512. In other words, there are 512 distinct values each for MVx and MVPx.

The difference ΔMVx (MVx – MVPx) can be in the range (-128, 127.5). Therefore, the alphabet size for ΔMVx is $2^{10} - 1 = 1023$. However, using rollover arithmetic, 9 bits of precision is sufficient to transmit the difference signal, in order to uniquely recover MVx from MVPx.

5   Let MVx = -63 and MVPx = 63 with K = log2(512) = 9. At quarter-pixel motion resolution, with an alphabet size of 512, the fixed point hexadecimal representations of MVx and MVPx are respectively 0xFFFFFF04 and 0x0FC, of which only the last 9 bits are unique. MVx - MVPx = 0xFFFFFE08. The differential motion vector value is:

$$\Delta\text{MVx} = \text{Rollover (0xFFFFFE08, 9)} = \text{0x008}$$

10   which is a positive quantity, although the raw difference is negative. On the decoder side, MVx is recovered from MVPx:

$$\text{MVx} = \text{Rollover (0x0FC + 0x008, 9)} = \text{Rollover (0x104)} = \text{0xF..F04}$$

which is the fixed point hexadecimal representation of -63.

The same technique is used for coding the Y component. For example, K is set

15   to 8 for the baseline MV range, at quarter-pixel resolution. In general, the value of K changes between x- and y-components, between motion vector resolutions, and between motion vector ranges.

IX.   Extensions

20   In addition to the embodiments described above, and the previously described variations of those embodiments, the following is a list of possible extensions of some of the described techniques and tools. It is by no means exhaustive.

1. Motion vector ranges can be any integer or fixed point number, with rollover arithmetic carried out appropriately.

25   2. Additional motion vector modes can be used. For example, a 4MV, 1/8-pixel resolution, six-tap interpolation filter mode, can be added to the present four modes. Other modes, including different combinations of motion vector resolutions, filters, and number of motion vectors, can also be used. The mode

- 41 -

may be signaled per slice, group of pictures (GOP), or other level of data object.

3. For interlace field-coded motion compensation, or for encoders/decoders using multiple reference frames, the index of the field or frame referenced by the motion compensator may be joint coded with extended motion vector information.

4. Other descriptors such as an entropy code table index, fading parameters, etc. may also be joint coded with extended motion vector information.

5. Some of the above descriptions assume a 4:2:0 or 4:1:1 video source. With other color configurations (such as 4:2:2), the number of blocks within a macroblock might change, yet the described techniques and tools can also be applied to the other color configurations.

6. Syntax using the extended motion vector can be extended to more complicated cases, such as 16 motion vectors per macroblock, and other cases.

Having described and illustrated the principles of our invention with reference to various embodiments, it will be recognized that the various embodiments can be modified in arrangement and detail without departing from such principles. It should be understood that the programs, processes, or methods described herein are not related or limited to any particular type of computing environment, unless indicated otherwise. Various types of general purpose or specialized computing environments may be used with or perform operations in accordance with the teachings described herein. Elements of embodiments shown in software may be implemented in hardware and vice versa.

In view of the many possible embodiments to which the principles of our invention may be applied, we claim as our invention all such embodiments as may come within the scope and spirit of the following claims and equivalents thereto.